Database Management System Architectures

This module is the third of four modules that describe the use of the Trusted Database Interpretation (TDI) of the Trusted Computer System Evaluation Criteria (TCSEC) for database product evaluation and certification. The basic terms and concepts presented in the TDI are summarized in TDI Module One. TDI Module Two describes various security policies which can be supported by a trusted database management system (DBMS). This module describes various architectural approaches for building a trusted DBMS. TDI Module Four describes other database security issues that are not covered by the TDI but are important issues in database security, such as inference, aggregation, and database integrity.

Module Learning Objectives

TDI Module One described approaches for evaluating a trusted DBMS while Module Two described the policies and requirements that the DBMS must enforce or meet. This module presents alternative architectures for a trusted DBMS and describes how these approaches affect the evaluation process and the DBMS's enforcement of its security policy. Upon completion of this module the student should:

1) be familiar with various architectural approaches for trusted DBMSs.

2) understand the pros and cons of each architectural approach with respect to an evaluation against the TDI.

3) understand the reference validation mechanism requirements for composed TCBs.

4) understand how security services can be distributed among the trusted DBMS and any underlying TCB (e.g., an underlying operating system (OS)).

Overview

A trusted DBMS must provide the same types of security services as a trusted OS. The differences lie in the underlying objects that the DBMS is protecting and the architectural alternatives that are available to DBMSs. Most of these alternatives rely on an underlying OS to provide basic services to the DBMS. Composed OS and DBMS TCBs must meet the same TCSEC requirements as that of a single OS TCB. The composition of two TCBs, each enforcing specific aspects of the overall security policy gives rise to various alternatives. The pros and cons of several database architectures and implementation alternatives are presented in the discussion below.

Architectural Approaches

There are a number of different approaches that can be taken to develop a trusted DBMS. In this section the following important examples are

considered: monolithic, TCB subsets, trusted subject, client-server, and integrity lock.

Monolithic

The monolithic architecture is a system that has a single TCB subset which enforces the system security policy and is self sufficient with regard to its needs and resources. It does not rely on other systems or TCBs to provide services it requires to perform its functions. From an evaluation perspective a system is considered monolithic when it is not feasible to divide its structure into multiple TCB subsets which can be separately evaluated. Of course, even with a monolithic approach, the internals of the system may be highly organized into separate modules, and in fact such structure is required for high assurance systems. This architecture is most likely to be used for a DBMS in the case where the DBMS runs directly on hardware instead of a general purpose operating system or in the case that the operating system (OS) and the DBMS vendor are one in the same.

Non-Monolithic

Monolithic DBMSs can be evaluated directly against the TCSEC. The other architectures discussed below all involve the composition of multiple TCB subsets playing a role in enforcing the system policy. An overview of this problem is given in [DTIN92]. The rest of this module assumes that the architecture and concepts being discussed are in reference to composite TCBs being evaluated using the TDI, except where specifically noted.

TCB Subsets Meeting the Conditions for Evaluation by Parts

The TDI defines TCB subsets and gives conditions required for evaluation of TCB subsets (i.e., evaluation by parts). The most common use of TCB subsets meeting the condition for evaluation by parts for DBMSs is a trusted DBMS implemented on top of a trusted OS. For this architecture to be eligible for evaluation by parts it must meet all the evaluation by parts requirements as defined in the TDI. In this case, the DBMS TCB subset directly depends on the OS TCB subset, and hence the DBMS TCB subset is less primitive than the OS TCB subset.

Of particular note is the requirement that each TCB subset includes all its trusted subjects. This means the OS TCB subset must include all its trusted subjects. The DBMS TCB subset must be untrusted with respect to the policy enforced by the OS TCB subset in order to meet the requirements for evaluation by parts. If the OS enforces a standard Mandatory Access Control (MAC) policy, the DBMS in this case cannot be trusted with respect to MAC.[1] The DBMS can enforce a DAC policy beyond that enforced by the OS.

---

[1] The following discussion is predicated on the assumption that the DBMS is privileged with respect to the MAC policy. It is also possible for the DBMS to be privileged with respect to the DAC policy, but this is not common practice.

Moreover, the combined DBMS and OS TCB subsets can enforce a MAC policy on DBMS objects such as tuples. However, this is done by storing tuples of different levels in OS objects of the corresponding levels. This is necessary since the DBMS is not trusted to violate the MAC policy enforced by the OS. The DBMS must operate as an untrusted subject at a level corresponding to the user's current working level and constrained to read only objects at levels dominated by the current level and write to objects at levels dominating the current working level. The DBMS can open for reading files at levels strictly dominated by the current working level and open files at the current level for reading and writing. In this architecture the enforcement of the MAC policy depends only on the correct operation of the OS TCB subset.

Among the other TCB subset conditions for evaluation by parts it is important to note that the OS TCB subset must provide support for the Reference Validation Mechanism arguments for the DBMS TCB subset. In particular, OS mechanisms must be used to guarantee that the DBMS TCB subset software and data cannot be tampered with by other processes running on the system, and the data in the database cannot be accessed except through the DBMS TCB subset. For example, the OS TCB subset file access control mechanism must protect the DBMS application file and all database files from direct access by user processes.

[JLUN88] gives more detail on this architecture. Oracle System 7 also exhibits this type of architecture in one of its configurations. Oracle's other configuration exhibits the trusted subject architecture, which is described below.

TCB Subsets Not Meeting the Conditions for Evaluation by Parts

The TCB subset constraints are explained in TDI Module One. TCB subsets not meeting all the constraints are not candidates for evaluation by parts. These conditions are analyzed for TCB subsets not meeting the conditions for evaluation by parts in TC-6.4.1 of the TDI. Below is a discussion of the trusted subject architecture which violates condition 3, "Trusted Subjects Included".

Trusted Subject

Often a trusted DBMS is trusted with respect to the MAC policy enforcement of the underlying OS TCB. As with the TCB subsets architecture described above, the DBMS depends on the OS TCB subset and uses its functions. However, this architecture does not meet all the requirements for evaluation by parts because the OS TCB subset does not include all its trusted processes. In this case the DBMS TCB subset is a trusted subject. The DBMS TCB subset still uses many OS functions, uses OS objects to store data used by the DBMS and the data for the database, and relies on OS TCB mechanisms to support Reference Validation Mechanism properties for the DBMS TCB.

In this architecture, the DBMS TCB is trusted to violate the security policy (e.g., MAC) constraints on reading and writing. The DBMS TCB subset is responsible for enforcement of a MAC policy on the objects it controls. For

example, the DBMS may enforce a MAC policy on database tuples while storing all tuples in a single OS object but only allowing users to access tuples for which their current working level entitles them. The OS object used for storing the data must be protected from direct access by untrusted subjects, perhaps using an object at a level not available for user logons. In order to extract appropriate tuples from this object and make them available to users, the DBMS uses its ability to violate the OS MAC policy to write down to users in order to give them access to the data.

Since this architecture does not meet all requirements for evaluation by parts it is not possible to have the evaluation of the OS and DBMS proceed with the same degree of independence which is possible with TCB subsets which do meet the conditions for evaluation by parts. However, it should still be possible to reuse much of the evidence generated during the original evaluation of the OS TCB subset. The exact amount of the OS evaluation evidence which can be reused will depend on the particular DBMS implementation.

The Informix trusted DBMS product is an example of this type of architecture.

Other Architectures

Below is a discussion of several other DBMS architectures which do not fall cleanly into the sections above. Client-Server architecture may or may not be a candidate for evaluation by parts, depending upon the implementation. The Integrity Lock architecture, as well as its design flaw, is discussed here as an example of an architecture which will not be evaluated by NSA.

Client-Server

In a client-server architecture the DBMS is divided into clients which:

> interface with users and application programs, receive queries from users and applications, return results of queries to users and applications;

and servers which:

> store data, process queries transmitted by clients, and return results of queries to clients.

With respect to security enforcement, the client-server architecture can be implemented in several different ways. Since the client and server must communicate, the client and server or portions of them will have to cooperate as peers to transmit data at different security levels and security data such as labels. In this respect, the analysis of these portions of the client and server can proceed based on the TNI.

A server in a client-server architecture can be implemented as an untrusted server. In an MLS environment, an untrusted server would have to be implemented as replicated server processes because a single untrusted server process would not have the privilege to transmit and receive information of different sensitivity levels. To handle multiple sensitivity levels the server

would have to be replicated one per sensitivity level. An architecture could go even further and replicate the server once per active user.

A trusted server is normally implemented as one or more processes that have the privilege to violate the system security policy and are trusted to enforce those aspects of the security policy that are within its domain. If a trusted server runs on top of a separately evaluated operating system, it may be considered as an instance of TCB subsets meeting the conditions for evaluation by parts or a trusted subject architecture depending on whether the server is trusted with respect to the policy enforced by the operating system. It is also possible that a trusted server could be implemented as a monolithic architecture.

Clients can be implemented in trusted or untrusted versions in the same manner as servers can. Both trusted and untrusted architectures are equally likely for clients, much more so than the likelyhood of an untrusted server.

The Sybase product is an example of the client-server architecture with a Trusted Subject server TCB which can support trusted or untrusted clients.

Integrity Lock

The Integrity Lock approach is one of the earliest and simplest approaches devised for trusted databases. However, because of inherent design flaws, this approach is not very popular today, and NSA will not evaluate a trusted database that has been developed using the integrity lock approach. It is included here for historical interest only.

The integrity lock approach was first devised as a method to build a trusted DBMS out of a commercial DBMS without requiring extensive modifications to the DBMS. In fact, the premise was that no modifications to the commercial DBMS would be required. Integrity Lock works as follows:

All tuples (or elements, depending on the label granularity) are labeled with a sensitivity label indicating its sensitivity. All tuples are 'locked' with a cryptographic seal by an integrity lock mechanism before being sent to the DBMS for storage in the database. This cryptographic seal is computed over all the data stored in the tuple (or element), including its label. The seal is then stored with the data in the database. Thus, all data stored in the database is sealed with its sensitivity label. When queries are presented to the DBMS, it processes the query and returns the result. All results are filtered by the integrity lock mechanism to ensure that the user only receives data that is marked at the user's session level or below AND the mechanism ensures that the seals and/or data have not been tampered with deliberately or accidentally by the DBMS. This check for tampering is done by recomputing the cryptographic seal and comparing the recomputed seal with the value returned with the data and label. If the seals differ, the data is not returned to the user and appropriate actions are taken to notify security personnel that the data has been tampered with. Since only the integrity lock mechanism has the key

used to compute the cryptoseal, untrusted processes in the DBMS cannot alter the data or label and recompute the correct seal.

As one can see, all trust for this approach is placed in the integrity lock mechanism, not the DBMS. This mechanism is normally much smaller and simpler than the DBMS, making it much easier to gain assurance that the mechanism works. This minimization of the amount of work required to develop a 'trusted' database plus the ability to adapt an integrity lock mechanism to several DBMSs is what made this approach attractive.

However, this approach has a significant design flaw which prevents this architecture from meeting the security policy requirements of the TCSEC at any level that requires Mandatory Access Control (e.g., B1, B2, B3, & A1). Vast amounts of sensitive information can be signaled past an integrity lock mechanism. While the integrity lock mechanism ensures each record is appropriately labeled, sensitive information may be encoded by the commercial DBMS or its applications using techniques like varying the ordering of released records or alternating long and short records, etc. This covert channel can easily attain high bandwidths using sophisticated methods and cannot be stopped by the integrity lock mechanism. Even if one is not concerned about malicious software in the DBMS utilizing this signaling channel, the integrity lock mechanism must be implemented at a level so that the back end portion of the DBMS only returns atomic storage units (e.g., tuples or elements) to the integrity lock mechanism. If tuples which are returned to the integrity lock mechanism are based on comparisons or other DBMS operations, tuples returned at a low sensitivity level could easily reveal information about more sensitive tuples.

Reference Validation Mechanism Arguments

The fundamental concept of reference monitor and its implementation as a reference validation mechanism was presented in TCSEC Module Six. TDI Module One introduces the extension of this implementation to TCB subsets. Below is a discussion of the RVM arguments for both composed and stand alone TCBs.

Reference Validation Mechanisms in composed TCBs

For the purposes of this discussion, we will assume that a composed TCB is made up of a DBMS TCB and an underlying OS TCB. However, these arguments still hold for other composed TCBs which are formed from separate components.

The reference validation mechanism requirements of isolation, remaining non-by passable/always invoked, and verifiable must be provided by the composed OS/DBMS TCB as a whole. To meet these requirements, a trusted DBMS must be complete and verifiable in its own right (as required by the TCSEC) and must rely on the underlying OS to isolate it and its data from user processes. The underlying OS TCB must protect the DBMS, its data, and the running DBMS process from tampering and prevent all access to the database except through the DBMS TCB. The DBMS TCB itself must then validate all accesses

to the objects it protects. Depending on the architecture, this validation may or may not rely on security mechanisms provided by the underlying OS TCB (such as MAC).

There are various ways an underlying OS TCB can protect a DBMS it supports, including any DBMS user data and program data it stores or uses to process queries. Typically, the DBMS TCB runs as a process in application space. This process is protected from users by the standard process isolation mechanism used to isolate all processes from one another. The DBMS program file and any DBMS applications and data are usually stored as files in the underlying OS file system or in raw disk space. The static portions of this (i.e., the execution file, canned queries, fixed tables) can be protected from illegal modification in a number of ways. Static data can be:

1) stored at a level below any normal user to prevent the static data from being modified by an untrusted process. The MAC policy of 'no write down' provides a natural method for preventing unauthorized modification while providing read access to all users. This works well for the DBMS application file and any other static data, which should never be modified, but must be capable of being read by all users.

2) marked at a special DBMS (mandatory) category such that only the trusted DBMS process possesses the category. This prevents all normal users from reading the application and any other DBMS files. However, this means that the DBMS TCB process must be invoked with this category (so it can read the DBMS files). Because the DBMS TCB is executing with its special category, any communication between the DBMS and user processes must be considered a write-down (since the special DBMS category will have to be dropped during the communications between the DBMS and the user). This write-down capability requires a special OS privilege which violates one of the conditions for evaluation by parts.[2] Hence, the DBMS must be considered a trusted process, not a separate TCB subset.

3) associated with a special discretionary identity or group. Access to that identity or group is possessed only by the trusted DBMS. However, the known weaknesses of DAC [TCSEC Module Nine - DAC] make this alternative very risky. If the DBMS TCB is trusted to enforce MAC, then this alternative would not be allowed because the labels stored with the data would be insufficiently protected from tampering by users. If the DBMS TCB was not trusted with respect to the underlying OSs MAC policy, then this approach might be acceptable.

Modifiable DBMS data can be protected using methods Two and Three. These methods will not only protect the data from illegal modification, they will also protect the data from illegal reads as well, such as attempts to access the data directly through OS file system commands. These methods will force all access

---

[2] See pp. 14-15 of The Trusted Databased Interpretation.

attempts to go through the DBMS TCB. Method One should not be used because the DBMS TCB would have to do a write down for each modification and the data would not be protected from attempts to read the data directly.

Reference Validation Mechanism in stand alone (monolithic) TCBs

A DBMS can be implemented in a front or back-end server without an underlying OS. This approach makes the DBMS a stand alone (i.e., monolithic) TCB. However, the DBMS must still communicate with users who reside on clients of the DBMS. These clients are typically hosts with operating systems that are connected to the DBMS server through a direct communications channel or a network.

The need to protect the trusted DBMS and its data from untrusted processes may be simplified when the DBMS is running in an environment that does not directly support users. A DBMS server supports its clients by receiving requests from its clients (another host), processing the request, and then returning the results. The users are not directly running on the server machine. However, it is still the case that the DBMS TCB must meet all RVM requirements. Demonstration of RVM requirements in this case is done in accordance with standard TCSEC criteria.

The introduction of a communications channel between the client and the server raises other issues that must be addressed. If the client/server communicate across a network then network security issues such as identification, authentication, data integrity, label integrity and label interpretation issues must be addressed. These issues are described in TNI Modules One through Four, which describe the development and evaluation of trusted networking products.

Distribution of Services

An important aspect of the development of a composed TCB is determining how the services provided by the composed TCB are to be distributed among the TCB subsets that make up the whole. Some services which must be provided by the composed TCB may be performed by one TCB subset with the service made available to other TCB Subsets. An example is Identification and Authentication (I&A) where one TCB subset (i.e., the OS) identifies and authenticates each user and then passes the authenticated identity and other information such as user session level on to other TCB subsets on request. Other services may be performed jointly by several parts of the TCB. A typical example is auditing where each TCB subset generates its own audit data. This data may or may not be stored in a central repository for later review. In every case it is essential that the services provide all TCBs with consistent and accurate information about security critical items such as the current working level and the user associated with subjects. The following describes possible distribution of services between a DBMS TCB and an underlying OS TCB.

MAC

For mandatory access control, the main architectural issue is whether or not to rely on the underlying OS to provide MAC. The following presents some of the pros (+) and cons (-) associated with reliance on OS MAC:

+ High assurance is easier to achieve since the underlying OS TCB does not have to be altered and MAC does not have to be provided by the DBMS TCB. This makes the evaluation more straightforward and less trusted code is required within the DBMS.

+ Can propose a TCB Subset argument and then avoid reexamining the underlying trusted OS. The trusted subject approach may still be required if any MAC privileges are needed by the DBMS TCB. Unfortunately, MAC privileges are frequently required to perform certain trusted operations such as downgrading, reclassification, database maintenance, etc.

- May be lower performance because there must be at least one file per security level. Performance degradation is somewhat dependent on the size, and security mix of the data in the database. A large number of levels and categories will tend to slow down this approach. A large amount of data with few combinations of levels and categories will tend to be affected much less.

  To understand the efficiency concerns, one must understand the relationships that normally exist between DBMS objects (i.e., tables, tuples, elements) and the underlying OS objects (files, disk space). A trusted OS will provide MAC at the file or raw disk space level, one label per object. A trusted DBMS will normally 'provide' MAC at the tuple or element level. However, the DBMS stores these tuples or elements in underlying OS objects. Since labels are only provided one per file, then there must be at least one OS file for each sensitivity of data in the database. If the tuples or elements in a table have many different sensitivities, then they must be stored in many separate files in order to allows the OS to label them correctly. This can severely impact the performance of the DBMS when it is manipulating large amounts of data of different sensitivities.

The other alternative is to provide MAC in the DBMS and rely on the underlying OS for primitive file services only. This is a trusted subject approach since MAC is enforced by the DBMS TCB.

- It requires appropriate assurance for MAC separation by the trusted DBMS and may require reevaluating aspects of the underlying OS.

+ It may be more efficient since data of different sensitivities can be mixed within a single OS file. Efficiency again depends on the mix of data.

+ This approach also makes it much easier for the DBMS to provide trusted administration functions across levels since the DBMS itself is trusted to enforce the MAC policy.

DAC

DAC is usually provided by the DBMS TCB in order to provide DAC at the tuple, table, and database level, which are abstractions built using underlying OS objects. DAC is also typically provided on views, which are ways of looking at and combining data stored in the base relations. The variety of DAC access modes (select, insert, update, delete, etc.) also makes it difficult to rely on any underlying file level DAC. DAC is almost always provided by the DBMS TCB because of the granularity, overlap and variety of DAC that must be provided by a trusted DBMS.

A ramification of this is that as long as the DBMS TCB enforces any aspect of the security policy, including DAC, it must be trusted. However, a DBMS TCB can be considered a TCB Subset if the policy it enforces is built on top of the policy enforced by the underlying OS. It is considered a trusted subject if it is granted any privileges which would allow it to violate the policy enforced by the underlying OS. Please refer to TDI Module One and section TC-4 of the TDI for more information on TCB Subsets, Evaluation by Parts, and Trusted Subjects.

I&A

The TDI says that Identification and Authentication of users can be provided by either the underlying OS or the DBMS. Relying on the underlying OS is more convenient for the user since I&A is only required once. To accomplish this a trusted path must be in place between the TCB's so they can share I&A and clearance data (i.e., the DBMS must be able to establish a trusted path to the more primitive OS TCB to obtain true I&A data).

A DBMS TCB can also rely on its own I&A mechanism, or use an I&A subsystem. However, this approach has a number of drawbacks. It is less convenient for the user since I&A is required more than once. There are also potential consistency problems between the two I&A databases. How does the DBMS ensure that the information in its I&A database matches what is in the underlying OS database, particularly with regard to user clearance information? If the DBMS relies on the underlying OS for clearance information then this must be sent to the DBMS. It is also difficult for the DBMS to ensure that the person logging in to the DBMS is the same person that logged on to the OS (i.e., one could login to the OS as user A and then login to the DBMS as User B and the DBMS would not know the difference).

Audit

The underlying OS will generate an audit trail for the objects it protects. However, the DBMS TCB must still generate its own audit records because the granularity of the underlying OS audit trail is insufficient (i.e., only at the file level). It is also possible, depending on the DBMS architecture chosen, for the identity of user actions to be masked by the DBMS application itself (i.e., DBMS performs actions on user's behalf, OS audit indicates DBMS did something, not the user). Thus, the DBMS must generate audit records that indicate on whose behalf each action was performed, and the success or failure of these actions. This raises the issue of how to combine or otherwise deal with

multiple audit trails. A mechanism must be in place that provides this capability.

Required Operating System Services

An important factor for a DBMS vendor who is building a trusted DBMS on top of a trusted OS is the security services that the trusted OS provides. In order for the two TCBs to cooperate in a trusted fashion, the trusted OS must provide certain capabilities which the DBMS TCB can use to exchange and verify security information with the OS TCB. The specific services that the DBMS TCB requires depends on the DBMS architecture and what underlying OS security mechanisms, such as MAC or I&A, the DBMS relies on.

If the DBMS relies on the OS to do I&A, then the OS must be able to export I&A information about a user to the DBMS TCB in a trusted fashion. This I&A information needs to include the user's authenticated identity, current session level, clearance, and any OS privileges that the user may possess. If the DBMS does its own I&A, then it must still be able to share I&A information with the OS to ensure that the two I&A databases are consistent. This is especially important to ensure that identities stored in audit trails can be mapped from their DBMS identity to their OS identity.

If MAC is provided by the DBMS, then mechanisms must be in place to share label information between the TCBs to ensure that the meaning of the DBMS labels has a correct mapping to the OS labels. To shield the DBMS from the details of the OS labeling scheme, the OS should also provide the capability to perform compare functions on labels. Label comparison functions such as Greater Than, Greater Than or Equal, Less Than, Less Than or Equal, Equal, Not Equal, as well as the ability to translate labels to their human readable counterparts and vice versa. Other useful functions to provide would be a Greatest Lower Bound or Least Upper Bound function.

Regardless of the architecture and capabilities, it is also important for the OS to be able to communicate in a trusted manner the sensitivities of all the objects it protects and the user sessions currently executing. This allows the DBMS to store its data in appropriately marked objects and to understand the current session levels of the users it is supporting. Other capabilities may be desirable.

The point being made here is that in order for two TCBs to cooperate in a trusted manner, specific capabilities must be provided by both TCBs to allow them to communicate security information in a trusted manner. This is especially critical when a DBMS TCB is to be built on top of a previously evaluated OS TCB. If the OS TCB does not provide the necessary capabilities, then it may have to be modified to provide these capabilities. Any modifications will require that some or all of the OS TCB be reevaluated, which can entail considerable time and expense.

Required Readings

The required readings are supplied as part of the source material for the module. These readings, and the module overview, provide all the material covered by the module test questions.

DTNI91 National Computer Security Center, *Trusted DBMS Interpretation of the Trusted Computer System Evaluation Criteria* (TDI), NCSC-TG-021, Version-1, April 1991.

Part One (except for section TC-5) and parts 1 - 7 of Appendix B describe various issues and alternatives surrounding Trusted DBMS architectures and their approaches and how the selected architecture will effect an evaluation against the TDI.

TLUN92 Research Directions in Database Security, Teresa Lunt, Editor, Springer Verlag, 1992. [Chapters 2 and 6]

Chapter 2 discusses the SeaView MLS DBMS research project being done at SRI. SeaView uses a TCB Subset architecture to build an MLS DBMS on top of an MLS Operating System (GEMSOS) using the underlying OS to enforce the MAC policy. Particular attention should be spent on section 2.7 which describes the architectural approach.

Chapter 6 describes the approach and architecture for both versions of the Sybase Secure SQL Server. The B1 version runs on a B1 UNIX platform while the B2 version runs on bare hardware, both offering the same user and application interface.

Other Related Readings

JHIN86 Thomas Hinke, *"Secure Database Management System Architectural Analysis"*, Proceedings of the National Computer Security Center Invitational Workshop on Database Security, June 1986.

JHEN86 Ronda Henning and Swen Walker, *"Computer Architectures and Database Security"*, Proceedings of the National Computer Security Center Invitational Workshop on Database Security, June 1986.

JVAR91 Rammohan Varadarajan, *"An Overview of INFORMIX Online/Secure"*, 14th National Computer Security Conference, October 1991.

JGAR86 Cristi Garvey, *"Architectural Issues in Secure Database Management Systems"*, Proceedings of the National Computer Security Center Invitational Workshop on Database Security, June 1986.

JHIN89 T. H. Hinke, *"DBMS Trusted Computing Base Taxonomy"*, 3rd IFIP Workshop on Database Security, September 1989.

TLUN92 Research Directions in Database Security, Teresa Lunt, Editor, Springer Verlag, 1992. [Chapter 4]

DTIN92 National Computer Security Center, *The Design and Evaluation of INFOSEC Systems: The Computer Security Contribution to the Composition Discussion*, Mario Tinto, C Technical Report 32-92, June 1992.

JBUR86 R. K. Burns, "*Towards Practical MLS Database Management Systems Using the Integrity Lock Technology*", 9th National Computer Security Conference, 1986.

JGRA84 R. D. Graubart, "*The Integrity-Lock Approach to Secure Database Management*", 1984 IEEE Symposium on Security and Privacy, 1984.

JGRA85 R. D. Graubart, K. J. Duffy, "*Design Overview of Retrofitting Integrity-Lock Architecture onto a Commercial DBMS*", 1985 IEEE Symposium on Security and Privacy, 1985.

JGRA89 R. D. Graubart, "*A Comparison of Three Secure DBMS Architectures*", 3rd IFIP Workshop on Database Security, September 1989.

JKNO87 R. Knode, "*TRUDATA: The Road to a Trusted DBMS*", 10th National Computer Security Conference, 1987.

JLUN88 T. F. Lunt, et al, *"Final Report Vol. 1: Security Policy and Policy Interpretation for a Class A1 Multilevel Secure Relational Database System"*, Computer Science Laboratory, SRI International, Menlo Park, California, 1988.

JNOT86 L. Notargiacomo and J. P. O'Connor, *"Report on Secure Distributed Data Management System Research"*, Proceedings of the National Computer Security Center Invitational Workshop on Database Security, June 1986.

JROU87 P. A. Rougeau, E. D. Sturms, "*Sybase Secure Dataserver: A Solution to the Multilevel Secure DBMS Problem*", 10th National Computer Security Conference, September 1987.

JTRO86 P. J. Troxell, "*Trusted Database Design*", 9th National Computer Security Conference, 1986.